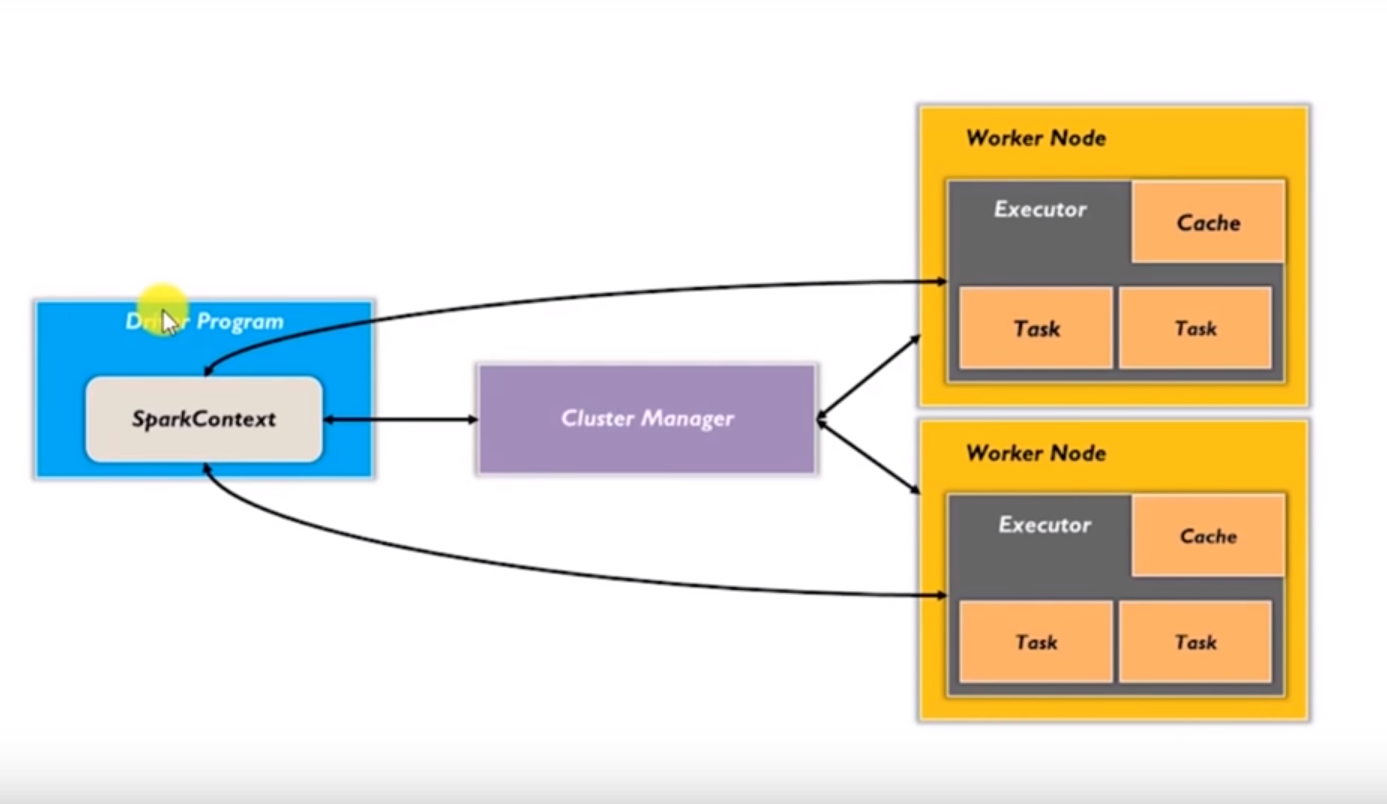
**Apache Spark**



**Spark architecture**

* Layered architecture (Master worker)
* Driver program basically drives the application (The code we have written is basically a driver)
* Driver programs is place where spark context is created (context is a gate to all functionalities of spark)
* Spark context along with driver takes care of job execution within the cluster
* Spark context works along with driver manager to manage jobs
* Cluster manager is responsible to acquire resources from cluster and allocate jobs
* Worker nodes are slave nodes whose job is to execute job operations
* Executor is a distributed agent is responsible for executing the tasks

So Spark context breaks the job in parts and distributes them to the worker nodes ,These task works on the partition RDD perform operation ,collect the results and return to main spark context.

Jobs can be executed parallelly among worked nodes.

**Spark supports below cluster :-**

**1.Spark standalone cluster**

**2.Apache Mesos**

**3.hadoop yarn**

**4.kubernatives**

**RDDs (Resilient distributed dataset):-**

To process data over multiple jobs we need to re-use and share the data, This is achieved through In memory data sharing (It’s a lot faster than network and data sharing) ,So RDD is way to achieve this in memory data sharing.

* Resilient means robust it can recompute any data in case of any node failure
* Its distributed among nodes.

**Data frames: -**It’s a distributed collection of data

* Data is organized under named columns (In form of table) which helps to filter, group, Aggregate
* Dataframe is untyped structured API in spark.
* Used along with spark SQL
* Construct data frames from structured data file ,RDDs, and storage like HDFS, HBase, Hive.
* Data frames are made on top of RDDs.
* Spark internally converts data Frames to RDDs.
* Its immutable in nature.
* Support multiple programing languages and Support multiple data sources.

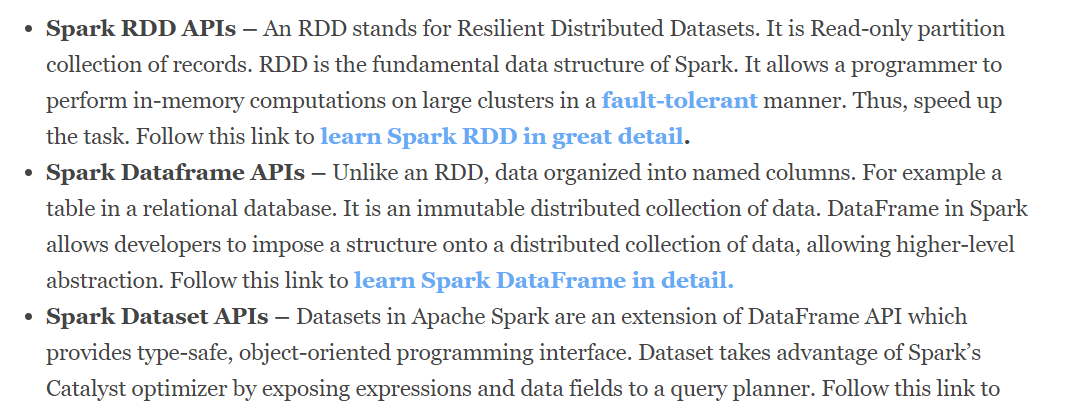
Elements within Dataframes are of Row type ,In spark row type cannot be parameterized by any type so We can say data frames are not typed and compiler can not check type at compile time.

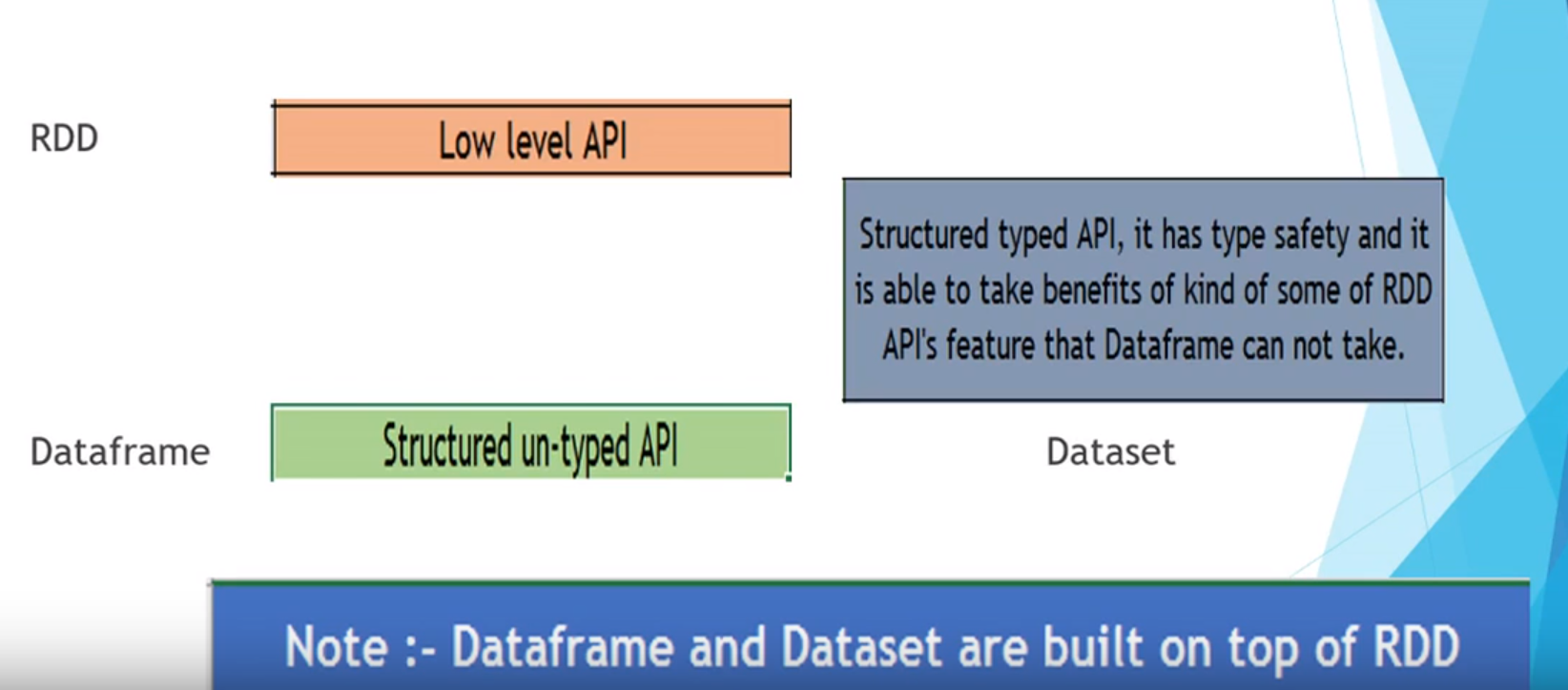
**Dataset**: -(Introduced in spark 1.6) It’s a data structure which is used to represent data which is need to be processed in spark

* Dataset also made on top of RDDs it basically provides best of both RDDs and Data Frames
* Data Frames had some limitation as with data frames you can not use functional programing

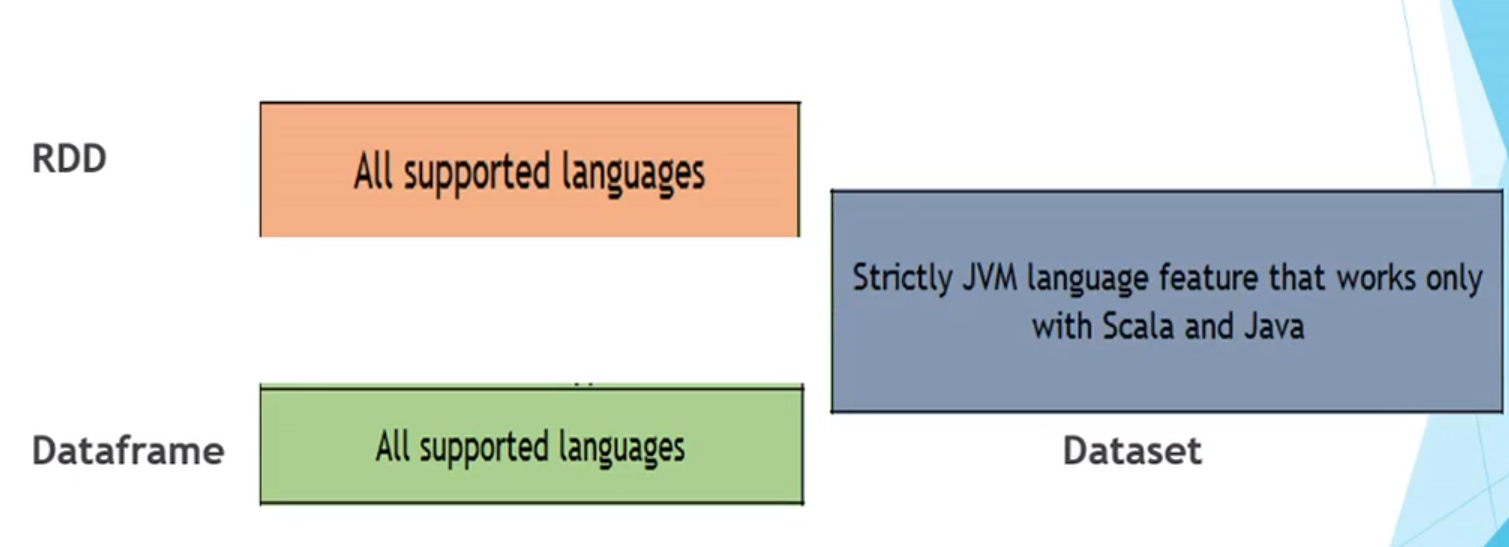
Methods like map functions.

* Dataset offer type safety (compile time type safety), as in case of data frames if you use wrong column name you wouldn’t know until you run it .
* Its immutable, once created you can’t edit it
* Strongly typed (compile safe).
* Its only available for Scala and Java
* Dataframe = Dataset[Row] ,It can be understand as Dataframes are also Dataset where object type is row. (Row is class in spark)
* Encoders -better Memory Utilization (Encoder basically maps domain specific object(Java beans) to sparks internal system)
* Datasets are faster and optimal as compare to data frames

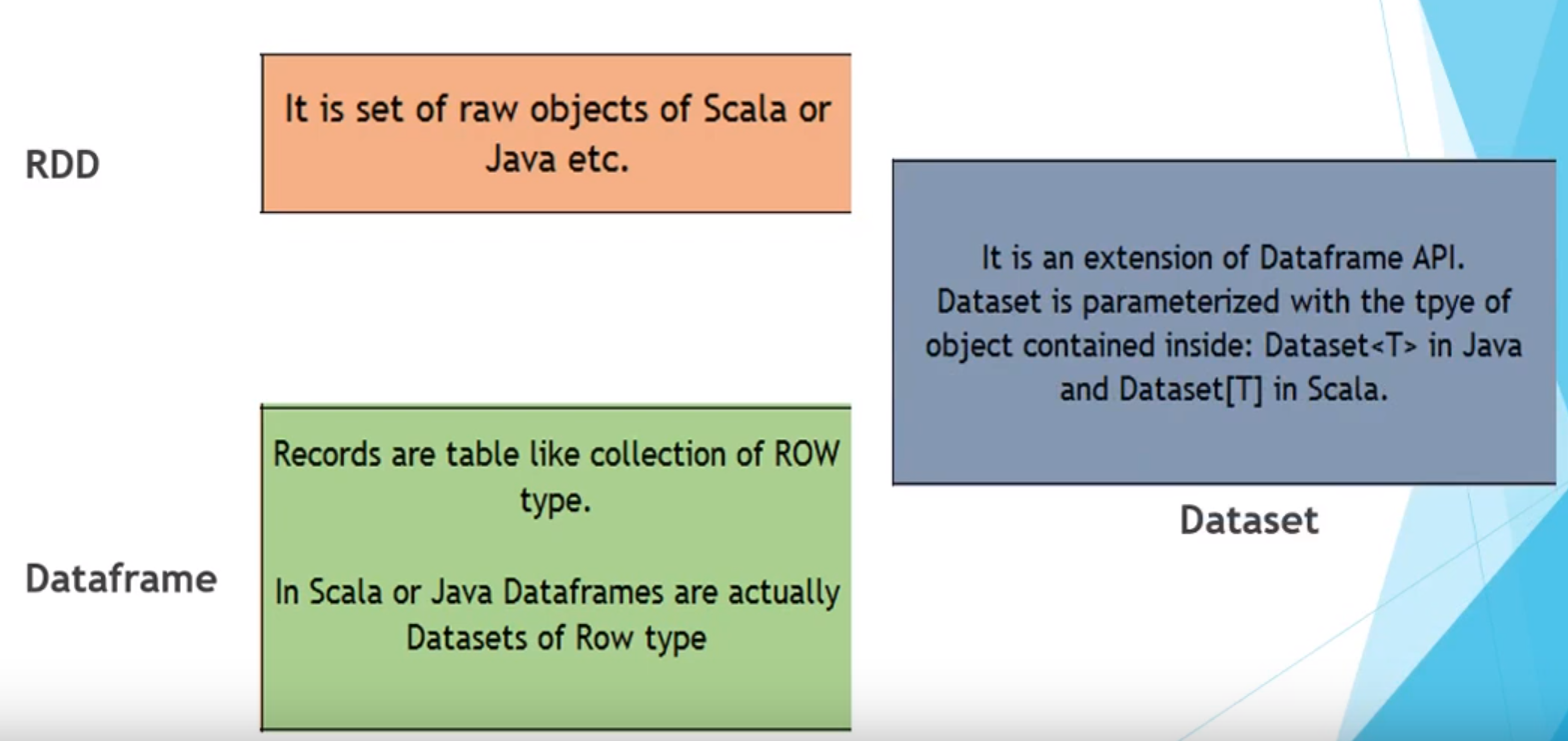




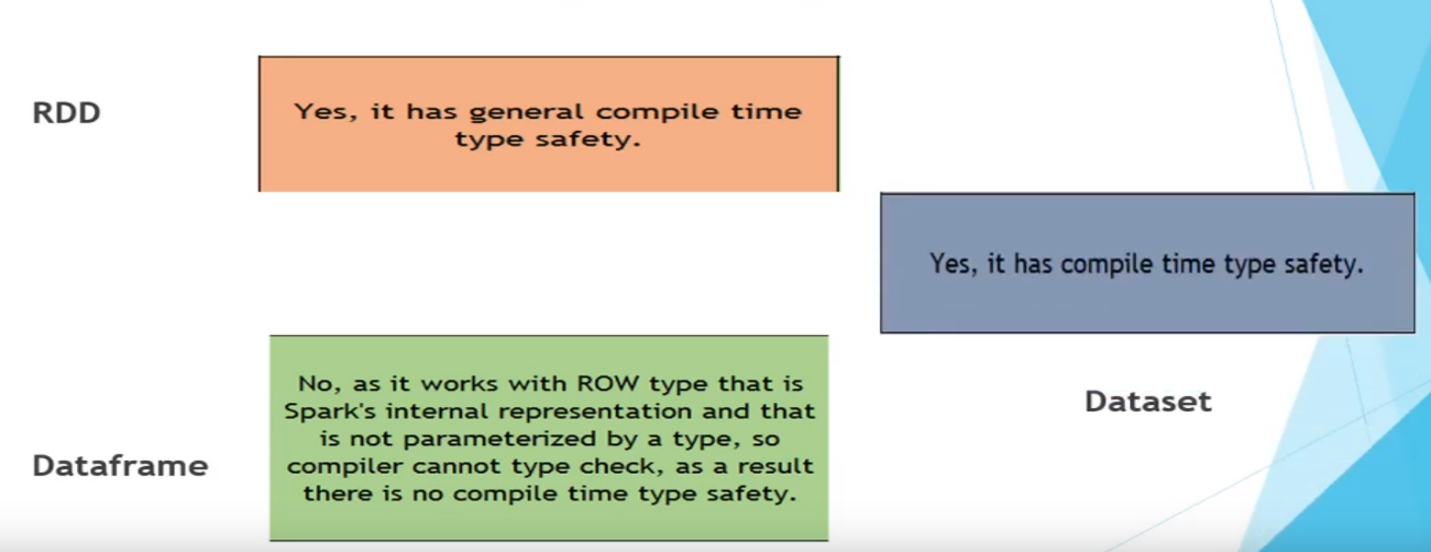
**Support for languages**



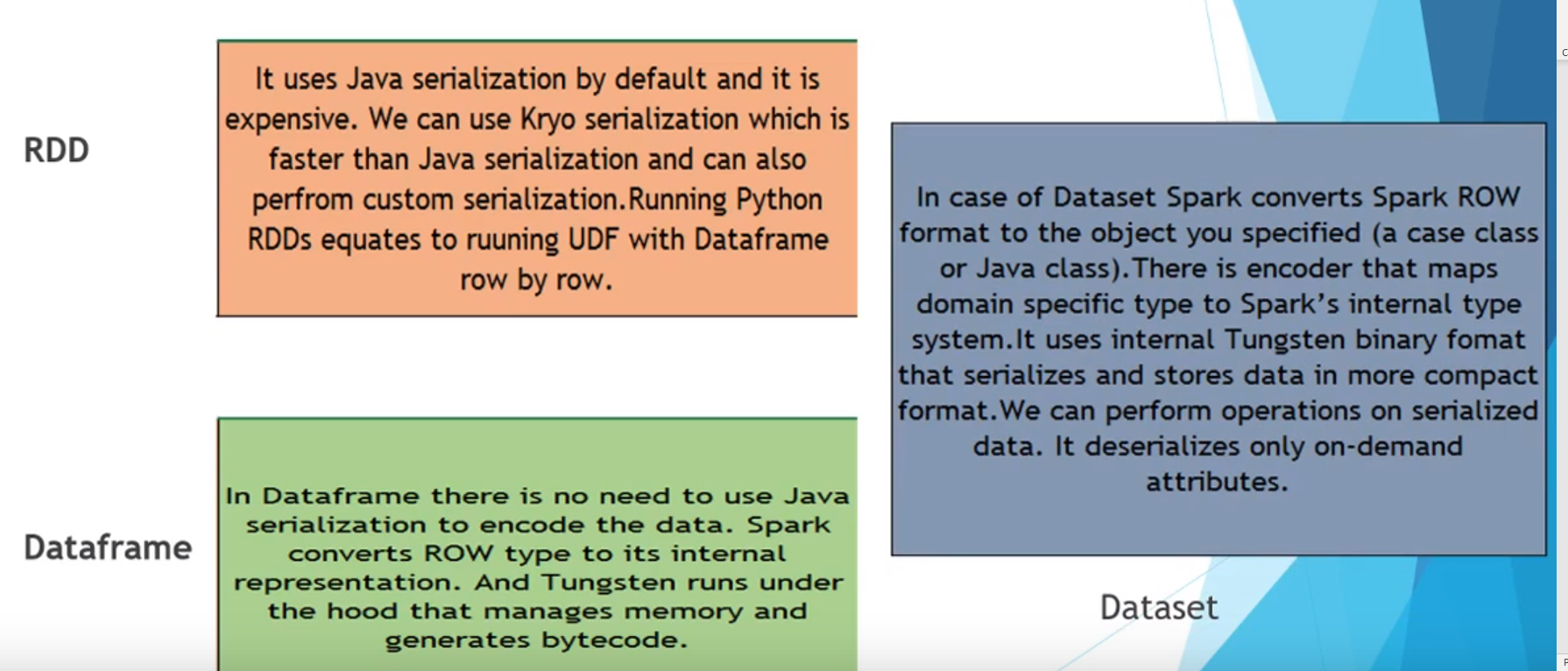
**Record Representation**



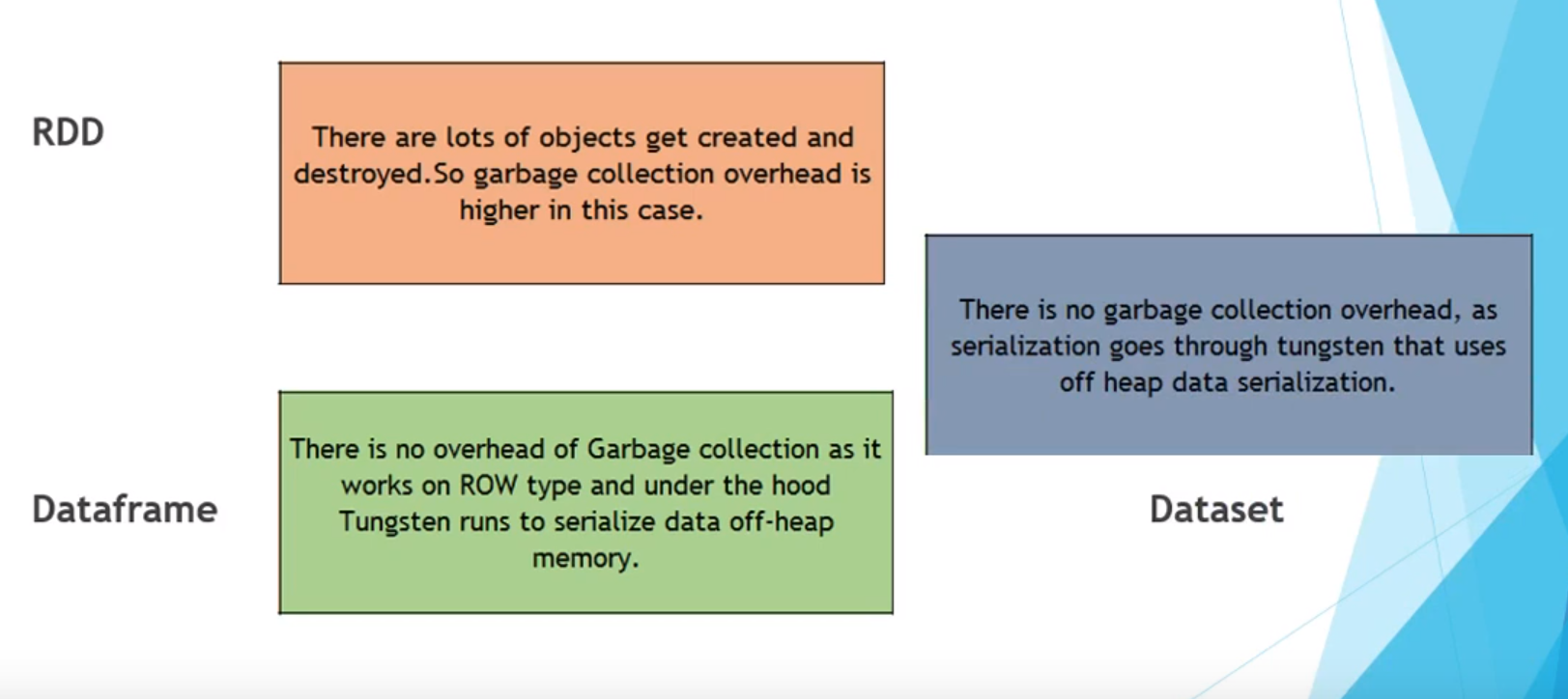
**Type Safety**



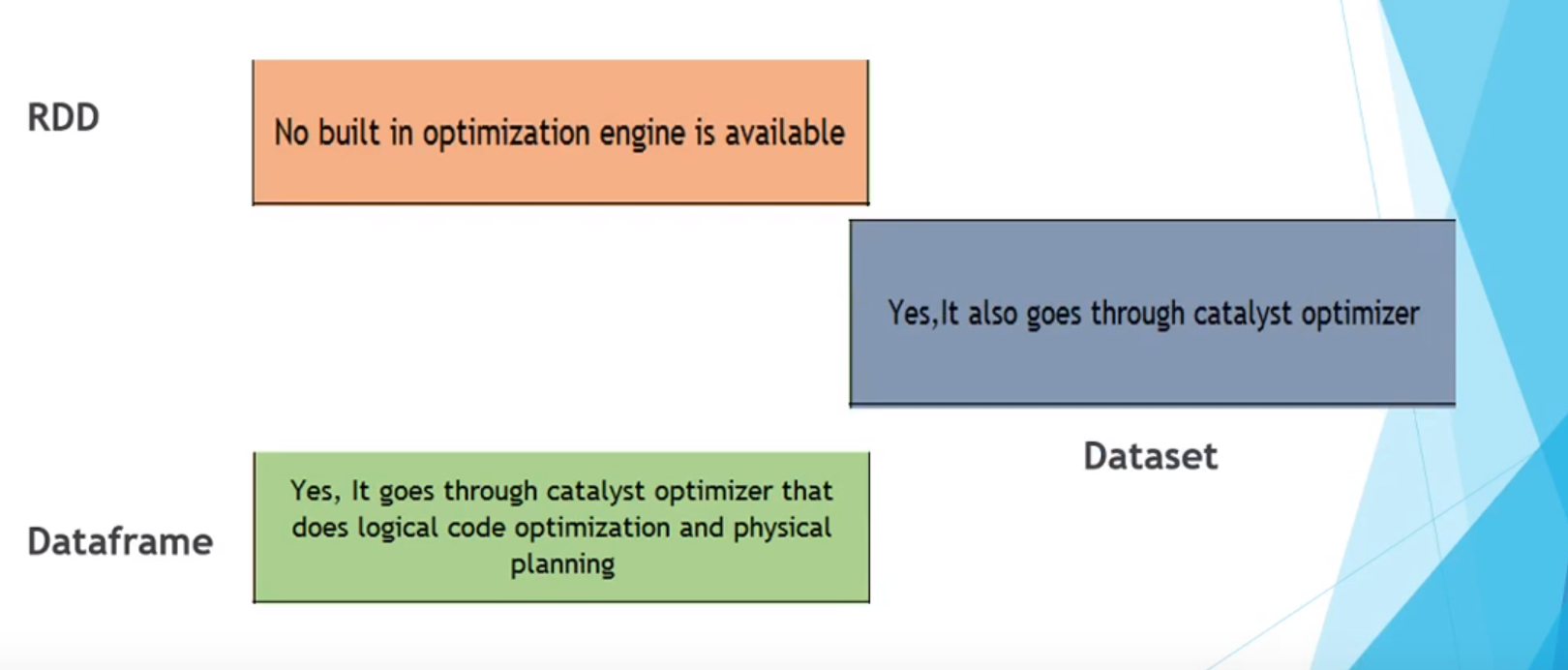
**Serialization**



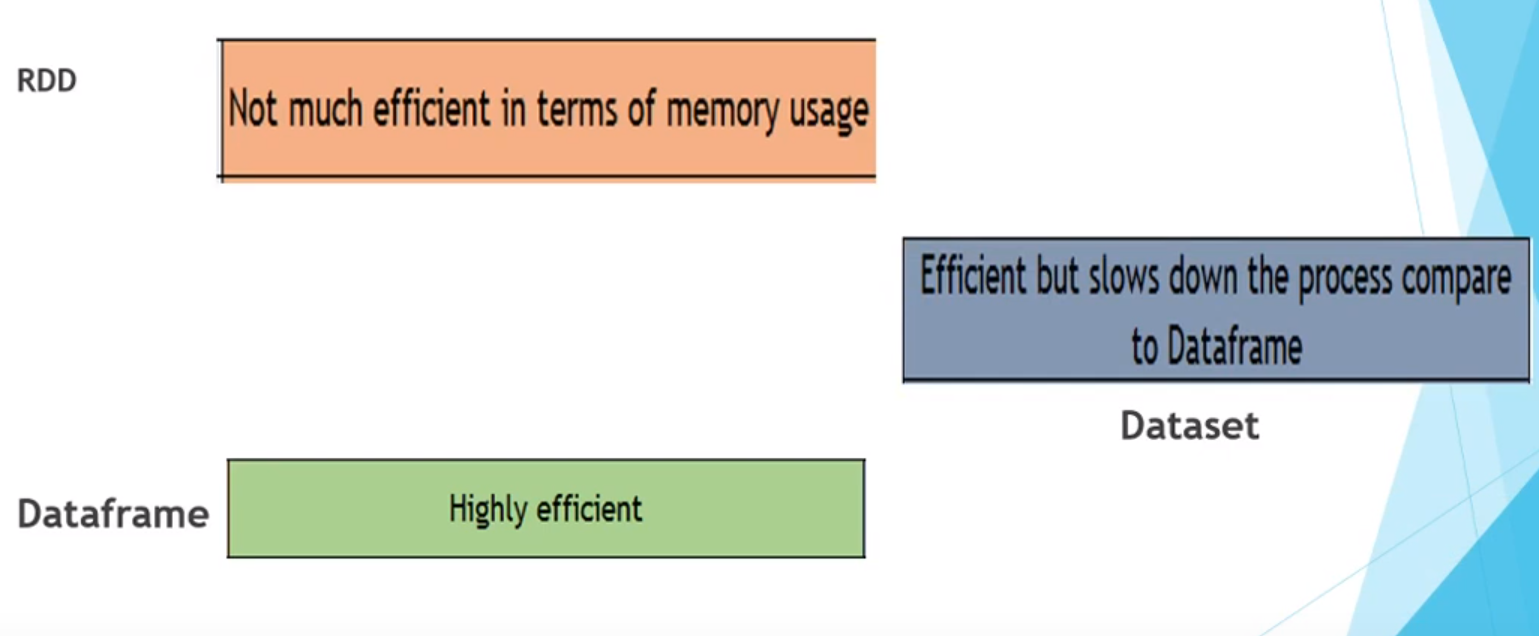
**Garbage Collection**



**Auto code optimization**



**Memory Efficiency**



**Sample Code with RDDs:-**

* As RDDs are low level API in spark we need Spark context to create RDDs

SparkConf sparkConf = new SparkConf().setMaster("local").setAppName("App name");

JavaSparkContext sc = new JavaSparkContext(sparkConf);

// read list to RDD

List<String> data = Arrays.asList("Learn","Apache","Spark","with","Tutorial Kart");

JavaRDD<String> items = sc.parallelize(data,1);

**Similarly, you can create RDDs from text file**

  // provide path to input text file

 String path = "data/rdd/input/sample.txt";

        // read text file to RDD

 JavaRDD<String> lines = sc.textFile(path);

        // collect RDD for printing

        for(String line:lines.collect()){

            System.out.println(line);

        }

**Sample Code with Dataset: -**

1. SparkSession spark = SparkSession.builder().appName("documentation").master("local").getOrCreate();
2. spark.sparkContext().setLogLevel("ERROR");
3. List<Row> list=**new** ArrayList<Row>();
4. list.add(RowFactory.create("one"));
5. list.add(RowFactory.create("two"));
6. list.add(RowFactory.create("three"));
7. list.add(RowFactory.create("four"));
8. List<org.apache.spark.sql.types.StructField> listOfStructField=**new** ArrayList<org.apache.spark.sql.types.StructField>();
9. listOfStructField.add(DataTypes.createStructField("test", DataTypes.StringType, true));
10. StructType structType=DataTypes.createStructType(listOfStructField);
11. Dataset<Row> data=spark.createDataFrame(list,structType);
12. data.show();
13. //Lets create the dataset of row using the Arrays asList Function
14. Dataset<Row> test= spark.createDataFrame(Arrays.asList(
15. **new** **Movie**("movie1",2323d,"1212"),
16. **new** **Movie**("movie2",2323d,"1212"),
17. **new** **Movie**("movie3",2323d,"1212"),
18. **new** **Movie**("movie4",2323d,"1212")
19. ), Movie.class);
20. test.show();

**sample code with DataFrames:-**

List<String> stringAsList = new ArrayList<String>();

stringAsList.add("buzz");

JavaRDD<Row> rowRDD = sparkContext.parallelize(stringAsList).map((String row) -> {

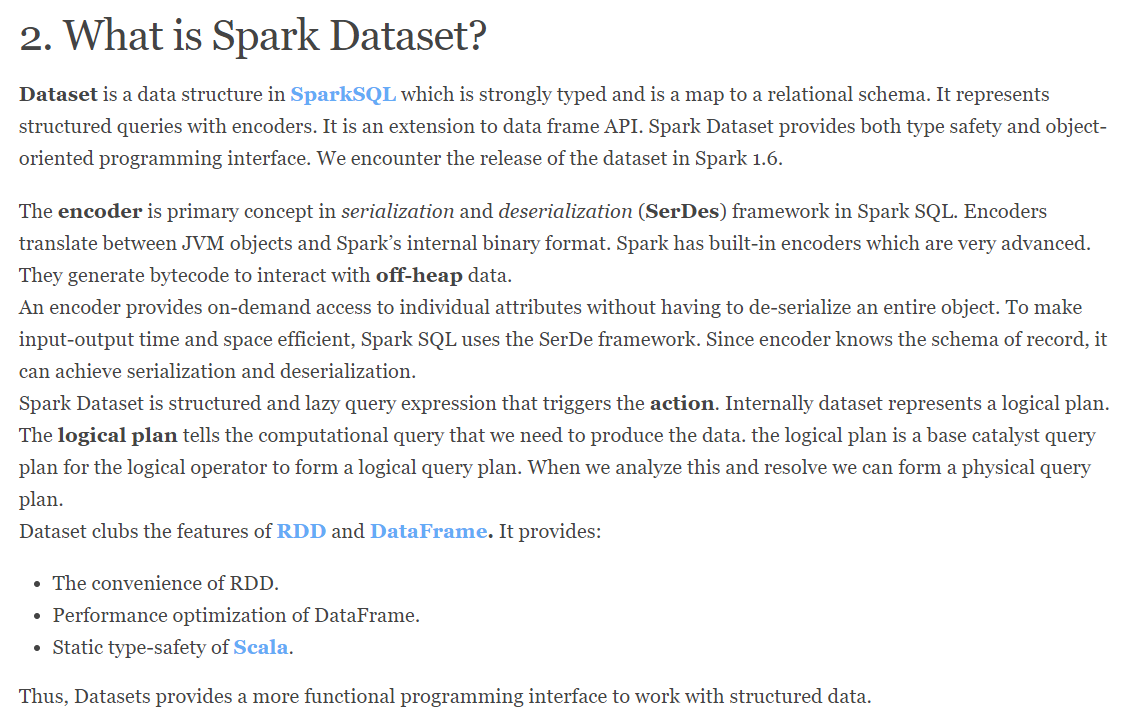
return RowFactory.create(row);

});

StructType schema = DataTypes.createStructType(new StructField[] { DataTypes.createStructField("fizz", DataTypes.StringType, false) });

DataFrame df = sqlContext.createDataFrame(rowRDD, schema).toDF();

df.show();



**Reference for code :-** <https://www.tutorialkart.com/apache-spark/spark-create-rdd-examples/>

<http://timepasstechies.com/create-spark-dataframe-java-list/>